The Tableau WorkBench

Pietro Abate

Research School of Information Science and Engineering
Australian National University

December 4, 2006

Outline

Outline

TWB At First Glance

- It is generic and extensible
- Modular framework to build theorem provers
- It is based on a general tableau algorithm
- Front-end for tableau calculi

Motto : Simple things should be simple, difficult things should be possible.

State of the Art

- Direct Provers
  - Fact/Fact++ (Description logics)
  - LWB (Modal logics, intuitionistic logics)
  - KSAT (modal logic)
  - Lotrec (arbitrary logics)
- Translational Provers
  - SPASS / MSPASS (first order logics, modal logics)
  - Vampire (first order logic)

Target Users

- Logicians without programming/technical background:
  - to experiment with new logics
  - to prototype theorem provers
  - to compare different technique using the same framework
- Automated reasoning teachers:
  - syntax similar to textbooks
  - no need to learn a programming language (for simple tasks)
- Expert automated reasoner with programming skills:
  - can be used as an embedded prover
  - the user level can be bypassed and each prover can be highly optimized

Outline

Tableau Methods

- Traditional tableau methods:
  - Modular
  - Efficient
  - Simple

  Traditional tableau calculi specify what it can be done, not how to do it.
- Extensions to the traditional formulation:
  - Histories (termination)
  - Variables
  - Tactic language

Outline

Definitions

A tableau rule **(name)** $\dfrac{n}{d_1 \ldots d_n}$ (side cond) is defined as:

- numerator, denominators
- principal formula, side formula
- each $n, d_1 \ldots d_n$ is a list of formula schema, e.g. $A\&B; X$, where A, B and X are meta variables
- side conditions (list of boolean functions)

A node is composed of a set of formula plus other data structures (Histories and Variables). A tableau for a set X is a tree of nodes with root X where the children of a node is obtained be instantiating a tableau rule.

- A node is partitioned according to a rule numerator
- A rule is applicable to a node if there exists a partition of the node and all side conditions are true.

- A branch in a tableau is a linear sequence of nodes
- A tableau is open if there exists a branch that is open
- The tableau method search for the existence of an open branch.

Rule Classifications

- Terminal Rule : **(name)** $\dfrac{n}{\perp}$
- Linear Rule : **(name)** $\dfrac{n}{d}$
- Universally Branching Rule: **(name)** $\dfrac{n}{d_1 \mid d_2}$
- Existentially Branching Rule: **(name)** $\dfrac{n}{d_1 \mid\mid d_2}$
- Conditionally Branching Rule: **(name)** $\dfrac{n}{d_1 \mid\mid\mid d_2}$

Removing Non-Determinism

- Node Choice: Out of many, we select a node to expand (the current node)
- Rule Choice: Given a node, we select a rule that is applicable to the current node
- Formula Choice: Given a node and a rule, we select which formula is the principal formula

How big ?

Core library by lines of code (loc)

component	loc
core	464
tableau library	818
syntax library	1624
data-type library	664
application (cli)	226
total	3800

How small ?

Logic modules by lines of code (loc)

logic	loc (tableau)	loc (functions)	total
CPL	18	98	126
K	21	72	93
KT	31	72	103
S4	41	72	113
PLTL	148	194	242

Outline

Tableau calculus for K

$$(\bot)\ \frac{\varphi; \neg\varphi; \Lambda}{\bot} \qquad (\wedge)\ \frac{\varphi \wedge \psi; \Gamma}{\varphi; \psi; \Gamma} \qquad (\vee)\ \frac{\varphi \vee \psi; \Gamma}{\varphi; \Gamma \mid \psi; \Gamma}$$

$$(K)\ \frac{\Diamond\varphi\ ;\ \Box\Sigma\ ;\ \Lambda}{\varphi; \Sigma}$$

Tableau calculus for S4 with histories

$$(\bot)\ \frac{\varphi; \neg\varphi; \Lambda \ :: \ \ldots}{\bot} \qquad (\wedge)\ \frac{\varphi \wedge \psi; \Gamma \ :: \ \ldots}{\varphi; \psi; \Gamma \ :: \ \ldots}$$

$$(\vee)\ \frac{\varphi \vee \psi; \Gamma \ :: \ \ldots}{\varphi; \Gamma \ :: \ \ldots \mid \psi; \Gamma \ :: \ \ldots}$$

$$(T)\ \frac{\Box\varphi; \Gamma \ :: \ \Pi, \Sigma}{\varphi; \Gamma \ :: \ \emptyset \ , \ \{\varphi\} \cup \Sigma}\ \varphi \notin \Sigma$$

$$(S4)\ \frac{\Diamond\varphi; \Lambda \ :: \ \Pi, \Sigma}{\varphi; \Sigma \ :: \ \{\Diamond\varphi\} \cup \Pi \ , \ \Sigma}\ \Diamond\varphi \notin \Pi$$

Rule Syntax (1/4)

```
CONNECTIVES
  And,  "_&_",  Two;
  Or,   "_v_",  Two;
  Imp,  "_->_", One;
  DImp, "_<->_", One;
  Not,  "~_",   Zero;
  Dia,  "Dia_", Zero;
  Box,  "Box_", Zero
END

HISTORIES
  (DIAMONDS : Set of Formula := new Set.set);
  (BOXES    : Set of Formula := new Set.set)
END
```

Rule Syntax (2/4)

```
RULE Id
{ a } ; { ~ a } ; x
===============
  Close
END
```

$$(\bot)\ \frac{\varphi; \neg\varphi; \Lambda}{\bot}$$

```
RULE And
{ a & b } ; x
==============
  a ; b ; x
END
```

$$(\wedge)\ \frac{\varphi \wedge \psi; \Gamma}{\varphi; \psi; \Gamma}$$

Rule Syntax (3/4)

```
 RULE Or
 { a v b } ; x
=================
   a ; x | b ; x
 END
```

$$(\vee)\ \frac{\varphi \vee \psi; \Gamma}{\varphi; \Gamma \mid \psi; \Gamma}$$

```
RULE K
{ Dia a } ; Box x ; z
---------------------
        a ; x
END
```

$$(K)\ \frac{\Diamond\varphi\ ;\ \Box\Sigma\ ;\ \Lambda}{\varphi; \Sigma}$$

Rule Syntax (4/4)

```
RULE S4
{ Dia a } ; z
----------------------
a ; SIGMA

COND notin(Dia a, PI)
ACTION [ PI := add(Dia a, PI) ]
END
```

$$(S4)\ \frac{\Diamond\varphi; \Lambda\ ::\ \Pi, \Sigma}{\varphi; \Sigma\ ::\ \{\Diamond\varphi\} \cup \Pi\ ,\ \Sigma}\ \Diamond\varphi \notin \Pi$$

Strategy

- $t = Skip$: t always succeeds
- $t = Fail$: t always fails
- $t = Rule(r)$: t succeeds when rule r is applicable.
- $t = t_1; t_2$: t fails if either t_1 or t_2 fail, succeeds otherwise.
- $t = t_1 | t_2$: t fails if both t_1 and t_2 fail, succeeds otherwise.
- $Repeat(t) = mu(X).(t; X | Skip)$: If tactic t succeeds, the tactic $Repeat(t)$ behaves like $t; Repeat(t)$. If t fails, then the tactic $Repeat(t)$ behaves like $Skip$. Repeat always succeeds.

Strategies for K and S4

- K : $Repeat(Repeat(And|Or|Id); K)$
- $S4$: $Repeat(Repeat(And|Or|T|Id); S4)$

Outline

Optimizations : Simplification

```
RULE And
{a & b} ; x
====================
a[b]; b[a] ; x[a][b]
END

RULE Or
{ a v b } ; x
===============================
   a ; x[a] | b ; x[b]
END
```

Optimizations : Simplification

```
let rec simpl phi a =
  let rec aux phi a = match a with
    |term (~ b) when b = a -> term(Falsum)
    |term (~ b)   -> term ( ~ [aux phi b] )
    |term (b & c) ->
              term ( [aux phi b] & [aux phi c] )
    |term (b v c) ->
              term ( [aux phi b] v [aux phi c] )
    |_ when phi = a -> term(Verum)
    |_ when phi = (nnf (term ( ~ a ))) ->
              term(Falsum)
    |_ -> a
  in boolean (aux phi a)
```

Optimizations : Simplification

$$\neg\varphi \vee \varphi \rightarrow \top \quad \varphi \vee \top \rightarrow \top \quad \varphi \vee \bot \rightarrow \varphi \quad \varphi \vee \varphi \rightarrow \varphi$$

$$\neg\varphi \wedge \varphi \rightarrow \bot \quad \varphi \wedge \top \rightarrow \varphi \quad \varphi \wedge \bot \rightarrow \bot \quad \varphi \wedge \varphi \rightarrow \varphi$$

$$\neg\bot \rightarrow \top \qquad \neg\top \rightarrow \bot$$

Benchmarks

s4 is the naive tableau prover for *S*4. The only optimization is caching;

s4c is as *s4* but without using caching;

s4s is as *s4* but using caching and simplification and semantic branching;

s4bj is as *s4* but using caching and back-jumping and semantic branching;

s4sbj uses all the previously mentioned optimization.

Benchmarks

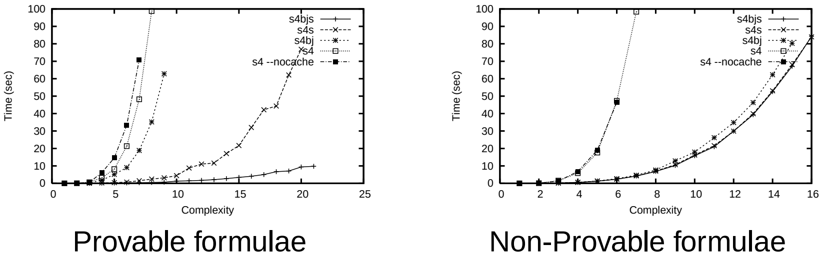
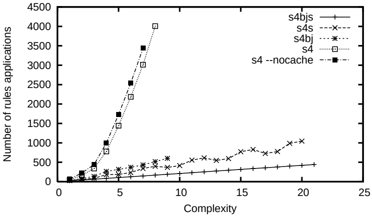
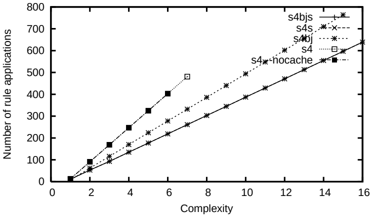


Table: S4 (time)

Benchmarks



Provable formulae



Non-Provable formulae

Table: S4 (rules)

Outline

Unified Branching Logic (UB)

- Introduced by Ben-Ari et al. in 1982
- Subsumed by CTL (Emerson et al. 1982)
- It extends linear time logic without considering the operator Until.

Syntax

$$
\begin{array}{rcl}
p & ::= & p_1 \mid p_2 \mid \cdots \\
\varphi & ::= & p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \bot \mid \top \\
 & & \mid AG\varphi \mid AF\varphi \mid AX\varphi \\
 & & \mid EG\varphi \mid EF\varphi \mid EX\varphi
\end{array}
$$

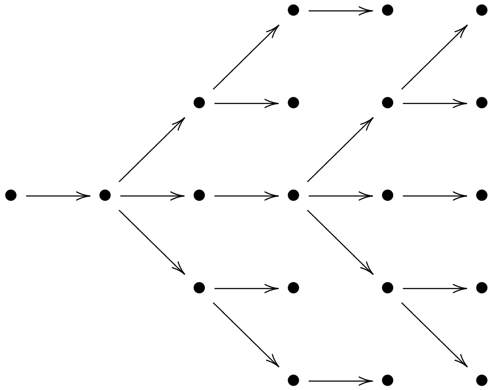
Semantics

Let $M = (S, R, L)$ be a *UB*-model. The truth value $M, s \Vdash \varphi$ of a formula φ in a state $s \in S$ is recursively defined as follows:

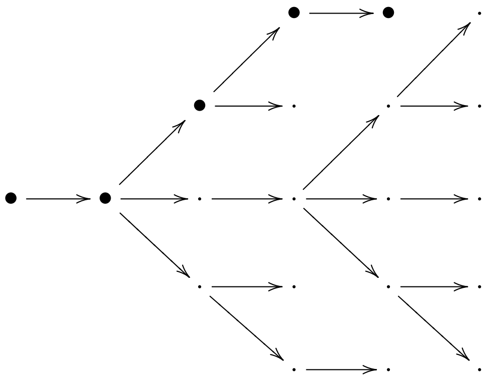
1. $M, s \Vdash p$ iff $p \in L(s)$
2. $M, s \Vdash \neg\varphi$ iff $M, s \nVdash \varphi$
3. $M, s \Vdash \varphi_1 \vee \varphi_2$ iff $M, s \Vdash \varphi_1$ or $M, s \Vdash \varphi_2$
4. $M, s \Vdash \varphi_1 \wedge \varphi_2$ iff $M, s \Vdash \varphi_1$ and $M, s \Vdash \varphi_2$
5. $M, s \Vdash EX\varphi$ iff $\exists t \in S$ such that sRt and $M, t \Vdash \varphi$
6. $M, s \Vdash AF\varphi$ iff $\forall$ fullpaths b with $b_0 = s$, $\exists t \in b$ such that $M, t \Vdash \varphi$
7. $M, s \Vdash EF\varphi$ iff $\exists$ a fullpath b with $b_0 = s$ and $\exists t \in b$ such that $M, t \Vdash \varphi$

The semantic of AX, AG and EG is given by duality with EX, EF and AF, respectively.

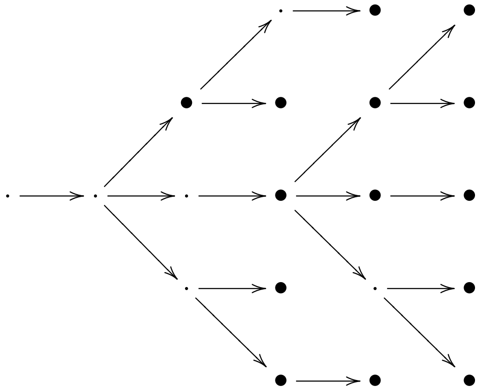
Intuition for AG



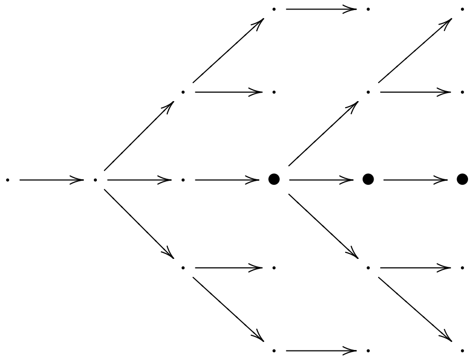
Intuition for EG



Intuition for AF



Intuition for EF



Linear Rules

$$(\wedge)\ \frac{\varphi \wedge \psi \ ;\ \Gamma :: \cdots}{\varphi \ ;\ \psi \ ;\ \Gamma :: \cdots} \qquad (D)\ \frac{\Gamma :: \cdots}{EX\top \ ;\ \Gamma :: \cdots}\ EX\varphi \notin \Gamma$$

$$(AG)\ \frac{AG\varphi \ ;\ \Gamma :: \cdots}{\varphi \ ;\ AXAG\varphi \ ;\ \Gamma :: \cdots} \qquad (EG)\ \frac{EG\varphi \ ;\ \Gamma :: \cdots}{\varphi \ ;\ EXEG\varphi \ ;\ \Gamma :: \cdots}$$

- The (D) enforces seriality
- The (AG) captures the axioms $AG\varphi \leftrightarrow \varphi \wedge AXAG\varphi$
- The (EG) captures the axioms $EG\varphi \leftrightarrow \varphi \wedge EXEG\varphi$

Universally Branching Rules

$$(EF)\ \frac{EF\varphi \ ;\ \Gamma :: Fev, Br :: uev}{\varphi \ ;\ \Gamma :: \{EF\varphi\} \cup Fev, Br :: uev_1 \mid EXEF\varphi \ ;\ \Gamma :: Fev, Br :: uev_2}$$

$$(AF)\ \frac{AF\varphi \ ;\ \Gamma :: Fev, Br :: uev}{\varphi \ ;\ \Gamma :: \{AF\varphi\} \cup Fev, Br :: uev_1 \mid AXAF\varphi \ ;\ \Gamma :: Fev, Br :: uev_2}$$

$$(\vee)\ \frac{\varphi \vee \psi \ ;\ \Gamma :: \cdots :: uev}{\varphi \ ;\ \Gamma :: \cdots :: uev_1 \mid \psi \ ;\ \Gamma :: \cdots :: uev_2}$$

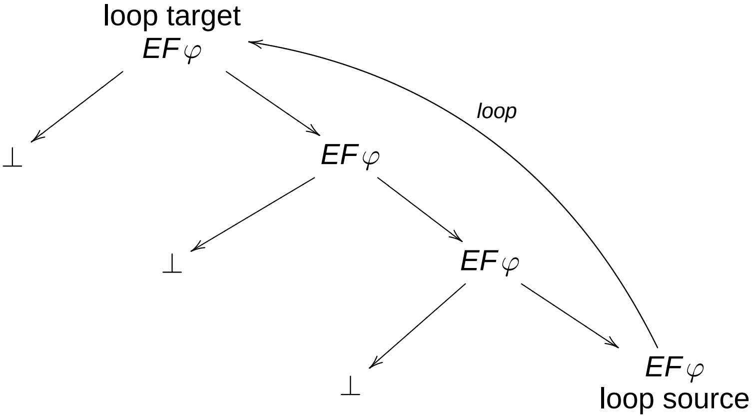
- The (EF) captures the fix-point $EF\varphi \leftrightarrow \varphi \vee EXEF\varphi$
- The (EF) captures the fix-point $AF\varphi \leftrightarrow \varphi \vee AXAF\varphi$
- The $(\vee)$ rule is standard except for computation of *uev*

Universally Branching Rules Conditions

where in the (EF), (AF) and $(\vee)$ rules :

$$
\begin{aligned}
f(x,y) &= \{i \mid (x,i) \in y\} \\
UEF &= \{(EF\psi, n) \mid f(EF\psi, uev_1) \neq \emptyset \;\&\; f(EF\psi, uev_2) \neq \emptyset \\
&\qquad \&\; n = min\ (f(EF\psi, uev_1)\ \cup\ f(EF\psi, uev_2))\ \} \\
UAF &= \{(AF\psi, n) \mid f(AF\psi, uev_1) \neq \emptyset \;\&\; f(AF\psi, uev_2) \neq \emptyset \\
&\qquad \&\; n = max\ (f(AF\psi, uev_1)\ \cup\ f(AF\psi, uev_2))\ \} \\
uev &:= \begin{cases} uev_1 & \text{if } uev_2 = \{(\text{false}, \_)\} \\ uev_2 & \text{if } uev_1 = \{(\text{false}, \_)\} \\ UEF \cup UAF & \text{otherwise} \end{cases}
\end{aligned}
$$

Unfolding the fix-point



Unfolding multiple fix-points

$$
\begin{tikzcd}
 & & \begin{array}{c} EF\varphi; \\ EF\psi \end{array} \arrow[dl] \arrow[dr] & & \\
 & \begin{array}{c} \varphi; \\ EXEF\psi \end{array} \arrow[dl] \arrow[d] & & \begin{array}{c} EXEF\varphi; \\ \psi \end{array} \arrow[d] \arrow[dr] & \\
\begin{array}{c} \varphi; \\ \psi \end{array} & \begin{array}{c} \varphi; \\ EXEF\psi \end{array} & & \begin{array}{c} EXEF\varphi; \\ \psi \end{array} & \begin{array}{c} EXEF\varphi; \\ EXEF\psi \end{array}
\end{tikzcd}
$$

Intuition for the Universally Branching Rules Conditions

The criteria for computing *uev* are as follows:

- if either branch is "closed" then keep the other uev. If both are "closed" then the $uev = uev_1 = \{(\text{false}, \_)\}$.
- if either uev_1 or uev_2 is empty because it has no "procrastinators", then the numerator is empty because $UEF = UAF = \emptyset$, hence $UEF \cup UAF = \emptyset$.
- if both are not "closed" and both contain "procrastinators", then $UEF \cup UAF$ constructs their "intersection" (sic!).

The crucial point is to simply ignore the notion of "open" and "fulfilled" and consider only "procrastinators".

Existentially Branching Rules

$$(EX)\ \frac{\begin{array}{c} EX\,\varphi_1 \ ; \ldots ; EX\,\varphi_n \ ; \ EX\,\Gamma \ ; \ AX\,\Delta \ ; \ \Lambda \\ :: Fev\,, Br \ :: uev \end{array}}{\begin{array}{c} \varphi_1 \ ; \ \Delta \\ :: \emptyset, Br_1 :: uev_1 \\ \ddagger(1) \end{array} \ || \cdots || \ \begin{array}{c} \varphi_n \ ; \ \Delta \\ :: \emptyset, Br_n :: uev_n \\ \ddagger(n) \end{array} \ || \ \begin{array}{c} EX\,\Gamma \ ; \ AX\,\Delta \ ; \ \Lambda \\ :: Fev\,, Br \ :: uev_{n+1} \\ \dagger \end{array}}$$

Existentially Branching Rules Conditions

with :

$n \geq 1$

$\ddagger(i)$ is the condition $\forall j \,.\, 0 \leq j \leq len(Br) \Rightarrow \{\varphi_i\} \cup \Delta \neq Br[j].core$

$\dagger$ is the (blocking) condition $\forall \psi \in \Gamma \,.\, \exists j \geq 0$ s.t. $\{\psi\} \cup \Delta = Br[j].core$

and where for $1 \leq i \leq n$:

$$Br_i = (\{\varphi_i\} \cup \Delta, Fev).Br$$

$$l = len(Br) - 1$$

$$UEV = \bigcup_{j=1}^{n+1} uev_j$$

$$uev := \begin{cases} UEV & \text{if } (\text{false}, \_) \notin UEV \,\&\, \forall(\_, n) \in UEV \,.\, n \leq l \\ \{(\text{false}, l)\} & \text{otherwise} \end{cases}$$

Intuition for the Existentially Branching Rule Conditions

The criteria for computing *uev* are as follows:

- if the i^{th} branch is closed than $uev = \{(\_,\text{false})\}$.
- if any *uev* contains an iterated eventuality that loops lower then this point ($n \leq len(Br)$) then $uev = \{(\_,\text{false})\}$.
- if all eventualities loop higher than this point, then *uev* is equal to the union of all *uev*s.

Intuition for the Existentially Branching Rule

$$
\begin{tikzcd}
& \begin{array}{c} EX\,\varphi_1 \,;\, \ldots \,;\, EX\,\varphi_n \,; \\ EX\,\Gamma \,;\, AX\,\Delta \,;\, \Lambda \end{array} \arrow[dl, dashed] \arrow[d, dashed] \arrow[dr, dashed] & \\
\varphi_1 \,;\, \Delta \arrow[d, dashed] & \varphi_n \,;\, \Delta \arrow[d, dashed] & EX\,\Gamma \,;\, AX\,\Delta \,;\, \Lambda \arrow[ul, bend right, "\mathit{loop}"'] \\
{} & {} &
\end{tikzcd}
$$

Terminal Rules

$$(id)\ \frac{EX\,\Gamma\ ;\ AX\,\Delta\ ;\ \Lambda :: Fev, Br :: uev}{Stop}\ \{\neg p; p\} \subseteq \Lambda$$

where $uev := \{(\text{false}, len(Br))\}$

$$(block)\ \frac{EX\,\Gamma\ ;\ AX\,\Delta\ ;\ \Lambda :: Fev, Br :: uev}{Stop}\ \{\neg p; p\} \not\subseteq \Lambda \text{ and } \dagger$$

Block Rule Conditions

where $\dagger$ is the condition $\forall\psi \in \Gamma \,.\, \exists j \geq 0 \,.\, \{\psi\} \cup \Delta = Br[j].core$, and

$$
\begin{aligned}
Cores &= \{\{EF\varphi\} \cup \Delta \mid EF\varphi \in \Gamma\} \\
UAF &= \{\, (AF\varphi, i) \mid \exists\, c \in Cores, \exists\, i \text{ such that } c = Br[i].core \\
&\qquad \&\ \forall j \,.\, i \leq j \leq len(Br) \Rightarrow AF\varphi \in Br[j].core \\
&\qquad \&\ AF\varphi \notin Br[j].fev \,\} \\
UEF &= \{\, (EF\varphi, i) \mid \exists\, i \text{ such that } \{EF\varphi\} \cup \Delta = Br[i].core \\
&\qquad \&\ \forall j \,.\, i \leq j \leq len(Br) \Rightarrow EF\varphi \in Br[j].core \\
&\qquad \&\ EF\varphi \notin Br[j].fev \,\} \\
uev &:= \{(\psi, n) \in UAF \cup UEF \mid \psi \notin Fev\}
\end{aligned}
$$

Intuition for the Block Rule Conditions

A blocked iterated eventuality $(EV\varphi, n)$ is in *uev* iff

- The branch b from the loop target to the loop source is a procrastinator for $EV\varphi$.
- $EV\varphi$ is not fulfilled on b between the target and the source of the loop.

Soundness and Completeness

Proposition (Termination)

All UB-tableau for a node $\varphi :: \mathit{Fev}, \mathit{Br} :: \mathit{uev}$ always terminate.

Theorem

If $\mathcal{T}$ is an expanded tableau for $\varphi :: \mathit{Fev}, \mathit{Br} :: \mathit{uev}$, with $\mathit{uev} = \emptyset$ then φ is UB-satisfiable.

Theorem

If a formula φ is UB-satisfiable then there exists an expanded tableau for $\varphi :: \cdots :: \mathit{uev}$ where $\mathit{uev} = \emptyset$.

Outline

Conclusions

- The TWB is small compared with other theorem provers;
- It is generic and extendible;
- It can be used to experiment with new logics, but in principle also to build specialized TPs;
- It can be extended to incorporate well known optimizations.

Availability

- One The Web:

  `http://twb.rsise.anu.edu.au`
- Demo:

  `http://twb.rsise.anu.edu.au/twbdemo`
- Download (Via Darcs):

  `http://twb.rsise.anu.edu.au/Repository`